Task Timing for IRMs

Diagnostic tool for pSOS

Wed, Feb 11, 1998

As an aid to understanding how the tasks interact and are scheduled to run every 15 Hz cycle, an addition to the system code has been made that allows the optional recording of this detailed activity. This note describes how it was done.

The pSOS kernel permits the user to supply a task switch callout procedure (herein called Switch) that the kernel invokes as it prepares to switch from one task to the next. This mechanism has always been used by the system code to support turning on and off task LED lights as a real-time diagnostic suitable for use with a logic state analyzer. The code that operates the LED lights is very short and therefore does not affect the real-time performance of the system code adversely. (The same is true for the interrupt routine LED lights, where a single instruction suffices to turn a LED light on or off.) To support the new task timing recording feature, it is important to make it as efficient as possible, especially when the optional feature is disabled. To that end, the system checks for feature being enabled and sets a system global pointer variable accordingly.

To enable use of the feature, a data stream must be defined in the `DSTRM` table with the name "`TASKLOG `". It should be specified as having 16-byte entries and of type 1. The user portion of the header may be 8 for convenience of 16-byte entry alignment. The `DSTRM` entry should look something like the following:

```
0001 0010 0008 1000     qType=1, eSize=16, hSize=8, qSize=4K
0000 7000 0000 0000     Queue base address = 00007000
5441 534B 4C4F 4720     "TASKLOG " is name
0000 0000 0000 0000
```

At reset time, the data stream queue will be initialized, and the header should look like the following, after it has been running for some time (about one day):

```
0576:007000 0001 0010 0018 1000  qType=1, eSize=16, hOff=$18, qSize=4K
0576:007008 013F C889 0000 0000  Total #entries written = 20.9 x 10^6
0576:007010 0250 1000 0020 0000  IN offset=$250, limit=4K, start=$20
0576:007018 0000 0000 0000 0000
0576:007020 0080 0001 0000 0001  first entry
0576:007028 9802 1114 1750 100F  2/11/98 1417:50, cycle 10, 7.5 ms
0576:007030 0020 0000 0000 0000  second entry
0576:007038 9802 1114 1750 1010  2/11/98 1417:50, cycle 10, 8 ms
```

If the data stream is not defined at system reset time, the diagnostic record will not be written, with a minimal performance hit of 2 instructions to check the pointer.

Under operation with pSOS, Switch is invoked during task transition with a pointer to the old task control block (TCB) and a pointer to the new TCB. At reset time, the system initializes each TCB's user-private pointer field to point to the LED mask to be used for that task's LED lights, which is how the LED lights are supported. If the system global variable is non-NIL, its value is the address of the data stream queue header above.

The IN offset field in the queue header points to the current entry, which has already been filled except for the elapsed time field. Instead, the elapsed time field was initialized to the current timer count value during the previous invocation of Switch. Upon entry to Switch, upon finding the pointer to the queue header, the timer count value is read and the elapsed time computed, which is used to replace the value in the elapsed time field. Then the IN offset field is advanced to the next queue entry, and its fields are initialized. The structure of the entry is as follows, from the example above:

```
0080 0001 0000 0001  Queue Monitor Task. Elapsed time = 0.5 ms.
9802 1114 1750 100F  2/11/98 1417:50, cycle 10.
```

The first word is the task LED mask, which is a single bit set at the bit position given by the task number. Here are the LED masks and the associated tasks:

```
0000  Idle Task (when no other task needs to run)
0001  Classic Protocol Task
0002  Alarms Scanning Task
0004  Console Task (to process little console data or via Page G access)
0008  Page Application Task
0010  Small Memory Dump Task (last line of page display)
0020  Date and Time Task
0040  Update Task (updates data pool, sends replies, flushes net queue
0080  Queue Monitor Task (system housekeeping)
0100  Data Server Task (sends server replies)
0200  Serial Task (handles serial port input)
0400  Acnet Protocols Task
0800  D0 Protocol Task
1000  Acnet RETDAT/SETDAT Task
2000  SNAP Task (supports Internet Protocol family)
4000  spare
8000  spare
```

The second word is the value of the pending signals field in the TCB, but its significance during task transition is minimal. (The hope was that it would show the events that may have caused the task switch, but it apparently doesn't.) The next longword value is the elapsed time of task execution, which is simply how long it has been since that task was entered. The last 8 bytes specifies the time-of-day, where the final byte is the number of half-milliseconds since the start of the current cycle.

The means of timing task duration is different depending upon whether the system is running as an IRM, using a MVME-162 board, or as a local station, using a MVME-133 board. The IRMs use a microsecond counter for timing. The local stations use a half-millisecond counter for timing. In the above example, which ran on a local station, the elapsed time was `0001`, which just means the elapsed time was 0–1 ms. In this case, the time resolution is poor, but if one adds up the time from successive entries, the errors do not accumulate, since the timer counter is sampled only once in `Switch`.

Here is an example of a few entries from an IRM:

```
0004 0000 0000 0045  Console Task, 69 µs
9802 1115 0904 0350
0100 0000 0000 0040  Server Task, 64 µs
9802 1115 0904 0351
0000 0000 0000 6798  Idle Task, 26.5 ms
9802 1115 0904 0351
0080 0000 0000 0035  Queue Monitor Task, 53 µs
9802 1115 0904 0400
```

The Queue Monitor runs in response to the 15 Hz cycle interrupt that is received by the system that starts each cycle's activity. In this case, the last 26.5 ms of cycle 3 of 2/11/98 1509:04 required no useful task activity.